Evolvable Hardware for Space Applications

Adrian Stoica, Alex Fukunaga, Ken Hayworth, Carlos Salazar-Lazaro

Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Drive
Pasadena CA 91109, USA,
adrian.stoica@brain.jpl.nasa.gov
alex.fukunaga@jpl.nasa.gov
ken@brain.jpl.nasa.gov
salazc@brain.jpl.nasa.gov

Abstract. This paper focuses on characteristics and applications of evolvable hardware (EHW) to space systems, and describes research directions and ongoing work at JPL. Our motivation for looking at EHW is the need for more autonomous adaptive space systems. Even with a best design, incorporating the most advanced flight qualified technology at the time, and considering what is known about the environments it will operate in, during long missions the hardware becomes no longer optimal. On-board computers can be reprogrammed by uploading software, however the computing hardware becomes obsolete, and the sensing hardware may face situations outside its initial design specs, which makes the ideas of reconfigurable and evolvable hardware attractive. As safety of space systems is most critical, and inappropriate controls can be fatal, at this stage the application of space evolvable hardware is seen mainly in evolving adapted sensors and sensory information processing systems. The EHW directions explored at JPL include (1) intrinsic evolution on analog ASICs, which we demonstrated on a custom analog neural chip, (2) evolution of analog electronic circuits at CMOS transistor level, which is work in progress, (3) evolution of dynamical systems in state-space representations, for which a "modeling clay" approach was proposed, partly demonstrated in hardware, (4) evolution of algorithms for on-board signal processing, where we addressed lossless compression, for which we obtained compression ratios superior to that of the best lossless compression.

1 Space-oriented evolvable hardware

Spacecraft autonomy plays a key role in future space missions. An intelligent, autonomous spacecraft must be able to cope with problems for which solutions were not specified on ground, and should be able to adapt itself to new or changing environments. Ultimately, all adaptations originate in the on-board electronics that control such changes. Thus, it is important to address on-board electronics with the capability to evolve, i.e., modify itself to improve the performance of the systems it controls.

There are several aspects that need to be considered when addressing space-oriented EHW. A thing that we consider very important is a systems approach, understanding clearly that EHW is part of the bigger system for which an optimality is sought. We need to understand who/what provides the means for calculating a fitness function for candidate solutions, if there is a target functionality or reward mechanism stored in some memory on-board, or reinforcement comes from the environment. Then it is the issue if evolution is an alternative for providing a response in useful time. Also, of paramount importance is how safe an EHW is for the space system.

Given that for space systems (e.g. satellites, spacecraft, planetary probes or rovers) safety is most critical, at this moment we consider that EHW applications are more suitable to evolving adapted sensors and sensory information processing systems than for example trajectory control. All the operations from the moment signals reach the sensors until a decision based on the information it contains is made, or a coded signal is sent to ground, are fully inter-related and ultimately could be co-evolved in their ensemble to a global optimal signal processing efficiency. For simplicity, these are independently considered for evolution: operations that relate to signal acquisition (sensor evolution, involving modification of sensor sensitivity domain/profile, focus of attention, control of sensor arrays, etc.) signal pre-processing: (filtering, amplification), extraction of information for on-board decisions (such as sensor-pointing), or preparing a signal for transmission to earth (e.g. compression). The EHW directions we started exploring address each of the above operations. We performed experiments in intrinsic evolution on analog ASICs, trying to understand more about intrinsic EHW and integration of such chips into higher level systems such as control of sensor arrays, antennas and solar panels, instrument pointing (in this sense we evolved circuits with desired I/O characteristic function of one or multiple inputs). We address the evolution of electronic circuits, which can be used for filtering or other signal transformations, exploring the design of evolvable CMOS chips based on transistor and elementary circuit blocks (current mirrors, differential pairs, etc). We address evolution of complex dynamic systems, which can be used to learn decision mechanisms or system behaviors, such as coordinated etc. Finally we approached evolution of compression algorithms. On all the above we consider that adaptation by evolution is a promising direction, which we will continue to research.

We could categorize the EHW directions we currently pursue at JPL in:

(1) intrinsic evolution on analog ASICs, so far achieved on a custom analog neural chip
(2) evolution of analog electronic circuits at CMOS transistor level (not treated here)
(3) evolution of dynamical systems in state-space representations, for which a modeling clay approach was proposed, partly demonstrated in hardware
(4) evolution of algorithms for on-board signal processing, where we addressed lossless compression, for which we obtained compression ratios superior to that of the best lossless compression algorithms.

2. Intrinsic evolution on programmable analog ASICs

Thompson's successful intrinsic evolution on an FPGA [1] marked the first important milestone in a direction that promises certain advantages over extrinsic EHW. FPAA's lag behind their digital counterparts in terms of flexibility of programmability, but are rapidly becoming more suitable for intrinsic evolution, such that intrinsic evolution on general-purpose programmable analog devices will follow in a very near future.

Here, we report initial results on intrinsic evolution on dedicated (special purpose) analog chips (ASICs), more precisely analog neural chips. We are not aware of any previous published results on this subject, however, the domain of evolutionary neural networks [9] [10] [11] [12], as well as various analog neural chips exist for several years, so it is very possible that other researchers have already performed such experiments, without looking at them as intrinsic EHW but rather as neural "hardware-in-the-loop" evolutionary learning.

JPL has been involved for many years in the design of analog and digital concurrent ASICs, including a family of programmable analog neural network chips [2] [3] [5]. One of these chips, name-coded NN64, consists of 64 neurons, each with 64 digitally programmable synapses. The chip performs analog processing on analog input signals. The synapses have analog inputs received from chip inputs or from other neurons on the chip, which they multiply (using a multiplying DAC) with a digitally-stored weight, providing an analog signal to the neuron somatic level. At the somatic level the analog contributions of the synapses are summed and passed through a sigmodal non-linearity, providing analog neuron outputs. D/A and A/D data acquisition boards interface the chip to a PC, where all the algorithms and controls are run from LabView.

Signal processing from synaptic input to neural output takes ~250ns, while reprogramming the weights requires loading in rows of 64, 8 bits at a time (i.e. 64 clock cycles per neuron, random access to the neurons that need update) 64 rows for the full chip. Using a 33 Mhz clock cycle this would take less than 2 microseconds per neuron, and about 120 microseconds for the full chip; the speed in the current LabView setup where the download is controlled by software is about 3 orders of magnitude lower.

Test 1. The purpose of this test was to evolve a neural functional approximator. A feedforward three layer 5-3-1 network was used to learn a simple function of one variable. The target was a bell shape Gaussian curve. The genome was 23 bytes length, coding the values for the 23 8-bit synaptic weights. Each neuron was pre-biased to have a 2V output in the absence of the input signal. The fitness function was determined based on the sum of the squared errors between the calculated target function, and the circuit response, as measured at 15 input values. The algorithm ran for 160 generations each of 200 individuals. The result can be compared to the target in Fig. 1 (left). The response at a ramp signal is illustrated in the oscilloscope caption in Fig. 1 (right).

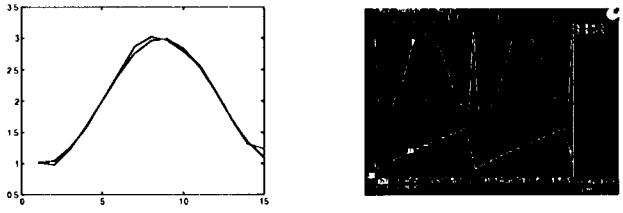

Fig. 1. A function learned by on the chip (intrinsic EHW): (left) closeness to target; (right) response on the oscilloscope.

Test 2. The purpose of this test was to evolve a visuo-motor controller for a mobile robot. The behavior was simple vision based track following, using one single neuron to map low resolution visual images to steering control. The inputs were 3x3 high-grain (low resolution) images, and the neuron output was a value which at its negative extreme mapped to complete steering left and at its maximum positive value mapped to complete steering right. A training set collected in a human-controlled driving session was simplified to obtain 12 training patterns like the ones illustrated in Fig. 2. The training set was stored in memory, evolution taking place without the robot in the loop. After 160 generations with a population of 200 the approximation error was below 5% on the training set, which proved sufficient for the evolved neural controller to drive the robot around the track.

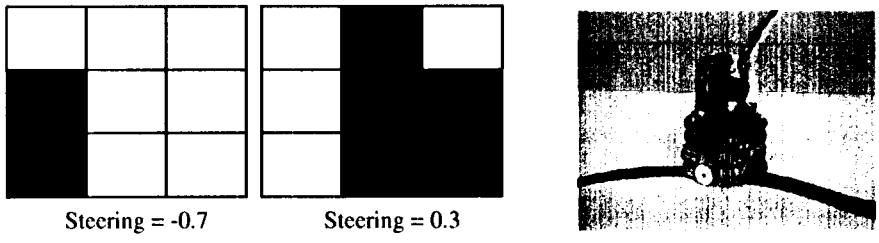

Fig. 2. Training sets used for learning and Khepera robot following a black trail painted on a white surface motor such that it follows a black track.

3. Evolution of dynamical systems in state-space representations : the modeling clay approach to EHW

The *behavior* of an electronic circuit can be described universally and precisely in terms of differential equations. Consider the state-space representation:

$$\dot{\vec{q}}(t) = \vec{f}(\vec{q}(t); \vec{x}(t))$$
$$\vec{y}(t) = \vec{g}(\vec{q}(t); \vec{x}(t))$$

where **x**(t) is a vector of continuous signal values coming into the system, **y**(t) is a vector of continuous output signal values, and **q**(t) is a vector of continuous internal state values, the "memory" of the system. The functions **f**() and **g**() are vector valued and in general non-linear. Figure 3 illustrates an example of the equivalence between a circuit in its schematic description and the state-space representation, graphically displayed by drawing the vector field **f**().

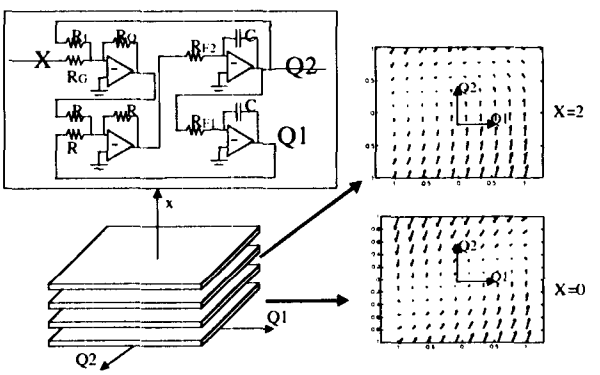

Fig. 3 **An active filter circuit from [6] and the three-dimensional state-space of the circuit dynamics with two Q1xQ2 vector field planes plotted at different points along the x-axis**

We have constructed a simple analog computer described in more detail in [8], which embodies the functionality:

$$\dot{q}_1 = f_1(q_1, q_2),\ \dot{q}_2 = f_2(q_1, q_2)$$

The functions used are bi-cubics, and the circuit is constructed such that the coefficients of the terms can be programmed digitally. Thus, a wide range of two state variable dynamic systems can be implemented on this computer. A special characteristic of this analog computer is that it is designed to be "context switchable". Each of the 20 coefficients of the cubic terms is obtained with a MDAC (Multiplying Digital to Analog Converter). Each of the 20 MDAC's are driven by a local digital memory store. This allows the entire analog computer's dynamics to change via a single broadcast addressing into the multiple MDAC memories. An analog storage stack could be included to store and pass the values of the analog state variables. The construction is supposed to mimic software process switching. Fast context switching allows the state-space of a dynamic system to be decomposed into a lookup-table of

smaller vector representations. Input, output, and internal state values remain analog. Taken together, this is an analog computer architecture that allows for virtual dynamic systems of any size and complexity limited only by memory size. The actual storage of the lookup-table is to be implemented using the CMAC [7] compression and smoothing algorithm. This method uses a combination of overlapping, averaged memories and hash-table storage in order to prevent the memory size from growing exponentially with the dimensionality of the state-space.

This architecture is designed as a very general piece of re-configurable analog hardware which possibly requires more memory and operates slower than an FPAA type solution, but allows "virtual circuits" of almost limitless complexity to be embodied. The key aspect of this context-switchable analog computer, which is desirable for evolvable hardware, is that its *native language* is the vector field representation of dynamic systems.

We are exploring, with this hardware architecture in mind, a novel EHW technique called the "modeling clay" approach to bio-inspired hardware [8]. This starts with the observation that biological evolutionary histories show progress toward complexity not through mutations that cause radical, qualitative changes in structure, but through *quantitative* changes in the percentage sizes and shapes of existing structure. Accumulation of these quantitative percentage size changes over many generations give rise to the complex qualitative changes of form and function. An excellent example of this phenomenon is the simulation, done by Nilsson and Pelger of the evolution of the vertebrate eye with graded index lens from only a flat layer of photosensitive cells with no optic structure[3]. Other examples abound in the paleontology record.

We have developed an approach for applying this analogy of cell population growth-based *physical* deformation to electronic circuit behavior. In the "modeling clay" approach, mutations are no longer thought of as changes in circuit connectivity or component values. The mutations now are non-linear stretches and molding of the vector field itself. Since the vector field represents more explicitly the "behavior" of the circuit, the hope is that allowing the evolutionary search algorithm to work with this behavioral description will allow much more efficient evolution of complex behavior.

4. Evolution of algorithms for on-board signal processing: results in lossless compression

Salami et al [13] have pioneered the application of EHW for image compression. In the context of space applications, compression is a very important problem because of the limited communications bandwidth between a spacecraft and the ground. Compression is necessary in order to enable the downlink of massive amounts of science data (images). Because image compression is extremely computationally intensive, a low-power, fast, hardware implementation of a compression algorithm is desirable. An EHW system could be used to automatically generate a hardware-based image compression algorithm specially adapted for the class of images captured by the spacecraft.

Both intrinsic and extrinsic EHW approaches are possible. For example, suppose a deep space probe needs to send thousands of similar images (e.g., atmospheric images) from the mission target (say, Pluto) back to Earth. The spacecraft could send several exemplar images back to the ground, where an FPGA configuration adapted for the class of images is evolved and uploaded to the spacecraft (extrinsic EHW). Alternatively, the spacecraft could evolve image-specific compression strategies directly using on-board hardware (intrinsic EHW).

To evaluate the utility of EHW for spacecraft on-board image compression, we have developed a genetic programming (GP) system to perform adaptive image compression based on predictive coding. Predictive coding uses a compact model of an image to predict pixel values of an image based on the values of neighboring pixels. A model of an image is a function model(x,y), which computes (predicts) the pixel value at coordinate (x,y) of an image, given the values of some neighbors of pixel (x,y), where neighbors are pixels whose values are known. Typically, when processing an image in raster scan order (left to right, top to bottom), neighbors are selected from the pixels above and to the left of the current pixel. To complete the compression, the error image (the differences between the predicted pixel value and the actual pixel value) is compressed using an entropy coding algorithm such as Huffman coding or arithmetic coding. If we transmit this compressed error signal as well as the model and all other peripheral information, then a receiver can reconstruct the original image by applying an analogous decoding procedure.

The GP system evolves s-expressions that represent nonlinear predictive models for lossless image compression. The error image is compressed using a Huffman encoder. Because the computational cost of evolving nonlinear predictive models using standard GP systems would be prohibitively expensive, we have implemented a highly efficient, genome-compiler GP system which compiles s-expressions into native (Sparc) machine code to enable the application of GP to this problem. The terminals used for genetic programming were the values of the four neighboring pixels Image[x-1,y-1],Image[x,y-1], Image[x+1,y-1], Image[x-1,y], and selected constant values: 1, 5, 10, 100. The functions used were the standard arithmetic functions (+,-,*, %), and MAX/MIN (which return the max/min of two arguments).

The system was evaluated comparing the size of the compressed files with a number of standard lossless compression algorithms on a set of gray scale images. The images used were science images of planetary surfaces taken from the NASA Galileo Mission image archives. The compression ratio of the following algorithms are shown in Table 1:

- evolved: the evolved predictive coding compression algorithm
- CALIC, a state-of-the art lossless image compression
- LOCO-I, recently selected as the new ISO JPEG-LS (lossless JPEG) baseline standard.
- gzip, compress, pack: These are standard Unix string compression utilities; *gzip* implements the Lempel-Ziv (LZ77) algorithm, *compress* implements the adaptive Lempel-Ziv-Welch (LZW) algorithm, and *pack* uses Huffman coding.
- szip, a software simulation of the Rice Chip, the current standard lossless compression hardware used by NASA.

It important to note that in our experiments, a different model was evolved for each image. In contrast, the other approaches (CALIC, GIF, etc.) apply a single model to every image. Thus, the time to compress an image using the genetic programming approach is several orders of magnitude greater than the time it takes to compress an image using other methods. However, the time to decompress an image is competitive with other methods.

Table 1. Compression ratios of various compression techniques applied to set of test images.

Image Name	Original size	evolved	CALIC	LOCO-I	com-press	gzip	pack	szip
Earth	72643	30380	31798	32932	42502	40908	55068	40585
Earth4	11246	5513	5631	5857	7441	6865	8072	7727
Earth6	20400	9288	10144	10488	11339	10925	13264	12793
Earth7	21039	10218	11183	11476	13117	12520	15551	13269
Earth8	19055	9594	10460	10716	11699	11350	13298	12465

The results obtained show that for science data images, an evolvable-hardware based image compression system is capable of achieving compression ratios superior to that of the best known lossless compression algorithms. In future work, we will focus on efficient, low-power mappings of evolved s-expressions to a general purpose FPGA on-board a spacecraft, as well as the evolution of models which perform well for a class of problems (as opposed to models specialized for individual images).

Acknowledgements

The research described in this paper was performed at the Center for Integrated Space Microsystems, Jet Propulsion Laboratory, California Institute of Technology and was sponsored by the National Aeronautics and Space Administration. The authors wish to thank Anil Thakoor, Sarita Thakoor, Benny Toomarian for ideas shared during discussions on evolvable hardware.

References

1. Thompson, A. et al.: Unconstrained evolution and hard consequences, in Towards Evolvable Hardware Sanchez & Tomassini eds., Springer-Verlag Berlin 1996 p136-165
2. Higuchi, T. et al.: Evolvable hardware and its applications to pattern recognition and fault-tolerant systems, in Towards Evolvable Hardware Sanchez & Tomassini eds., Springer-Verlag Berlin Heidelberg 1996 p118-135
3. Nilsson, D., Pelger, S.: A pessimistic estimate of the time required for an eye to evolve, Proceedings of the Royal Society of London, B, 256, p53-8
6. Horowitz, P., Winfield, H.: The Art of Electronics 2nd ed Cambridge Univ. Press 1989
7. Albus, J. S.: A new approach to manipulator control: the Cerebellar Model Articulation Controller (CMAC), Trans. of the ASME Sept.1975, p220-7

8. Hayworth, K., The "Modeling Clay" approach to bio-inspired electronic hardware, Submitted to ICES98.
9. Heistermann, J., The application of a genetic approach as an algorithm for neural networks, Lecture Notes in Computer Science v496: p297-301, 1991
10. Prados, D. L., New learning algorithm for training multilayered neural networks that uses genetic-algorithm techniques, Electronics Letters v28 (16): p1560-1561, July 30, 1992
11. Maniezzo, V., Genetic evolution of the topology and weight distribution of neural networks, IEEE Transactions on Neural Networks v5 (1): p39-53, Jan 1994
12. Yao, X., Liu, Y., A new evolutionary system for evolving artificial neural networks, IEEE Transactions on Neural Networks, v8 (3): p694-713, May 1997
13. Salami, M., Murakawa, M., Higuchi, T., Data compression based on evolvable hardware, Proc. Evolvable Systems Workshop, International Joint Conference on Artificial Intelligence, 1997